Autonomous Navigation Strategy for Orchards Relying on Sensor-based Nonlinear Model Predictive Control

Antoine Villemazet, Adrien Durand-Petiteville, Viviane Cadenat

Autonomous Navigation Strategy for Orchards Relying on Sensor-based Nonlinear Model Predictive Control

A. Villemazet[1,2], A. Durand-Petiteville[3] and V. Cadenat[1,2]

***Abstract*—This paper deals with autonomous navigation through orchards. It proposes a strategy based on a sensor-based model predictive control law coupled with a spiral-based framework allowing to deal in a similar manner with the two main tasks: the in-row navigation and headland maneuvers. The Model-based Predictive Control scheme allows to deal with stability, boundaries of the actuators, and offers a solution to switch between the different tasks. Simulation results based on ROS and Gazebo are presented at the end of the paper.**

I. INTRODUCTION

In this paper, we present a navigation strategy relying on sensor-based model predictive control to make a robot autonomously navigate in an orchard. This work belongs to the movement seeking at dealing with the need of doubling the worldwide food production by 2050 [1] via the mechanization of the agricultural tools [2]. More specifically, one focuses on the autonomous navigation challenge present in the mowing, spraying, pruning, and harvesting processes. To achieve these tasks, the robot has to autonomously drive from the entrance of an alley to its exit, navigate in the headlands to reach the entrance of the next row, and repeat these two steps to cover the whole area of interest. While navigation systems in open-field usually rely on GNSS receivers, this approach may fail in the orchards because of the tree canopy which may block the localization signal [3]. For this reason, orchard navigation strategies mainly rely on embedded exteroceptive sensors. Thus, the main approach to drive in the alleys consists in detecting the tree rows, computing geometrical lines in the robot's coordinate frame and using them for guidance. There exists numerous related works and a large spectrum of sensors has been considered, such as vision or Lidar. Some of them are briefly presented hereafter. In [4] and [5], a monocular camera is used as the main sensor. In the first one, a sky-based approach allows to compute the vehicle heading, while in the second one, a medial axis-based machine vision is designed to calculate the path to follow. In [6], the tree positions are computed by detecting their shadows in a point cloud provided by a RGB-D camera. Finally, in [7], a Time-of-flight camera allows to compute an occupancy grid of the alley. Lidars have also been used. In [8], the localization performances of a particle filter and a Kalman filter are compared, while in [9] lines, segments and circles are extracted to detect the tree rows. Both works rely on 2D Lidar data. Despite the variety of the approaches, none of the mentioned works deal with the headland navigation problem. Thus, robust, accurate and repeatable turning at the end of a row using relative positioning information with respect to trees is very difficult[10]. A first method proposed in [11] relies on dead reckoning with slip compensation to perform the u-turn. [12] follows a similar approach to execute maneuvers, despite the use of a 2D Lidar, a camera and a tree-detection algorithm to detect the end of the row. Another solution consists in introducing easily distinguishable artificial landmarks at tree rows extremities in order to detect the end of the current row, the entrance of the next one, and localize the robot during turning, using once again dead reckoning [13]. Finally, it was proposed to perform the headland maneuvers (u-turn in [14] and Ω-shape turn in [15]) by following a spiral centered on the last tree of the row. Its position is obtained thanks to a 2D Lidar and the spiral is tracked using a non-linear output feedback controller. These approaches thus allow to navigate in the headlands using exteroceptive data, thus increasing the robustness and accuracy of the maneuvers.

In this paper, we aim at continuing the works presented in [14] and [15]. First, we propose to extend the spiral tracking approach to the line following problem. Thus, both in-row and headland navigation rely on the same framework: a spiral tracking problem where the centers of the spirals are computed based on the position of the surrounding trees. Next, the non-linear output feedback controller is replaced by a Non-linear Model-based Predictive Controller (NMPC) to perform the spiral tracking. NMPC offers two advantages. First, it becomes possible to take into account several constraints. In this work, we include a terminal constraint to deal with the stability of the spiral tracking, which was proven to be only local in [14] and [15]. Moreover, we bound the control inputs to take into account the actuator limitations. In the future, additional constraints could be considered to deal with the presence of obstacles or the field of view of the sensors. Second, a NMPC scheme represents an interesting framework to deal with a sequence of tasks. Finally, we present a task switching method allowing to progressively modify the reference values in the prediction horizon. Thus, it becomes possible to switch from one task to another without creating discontinuities in the control law.

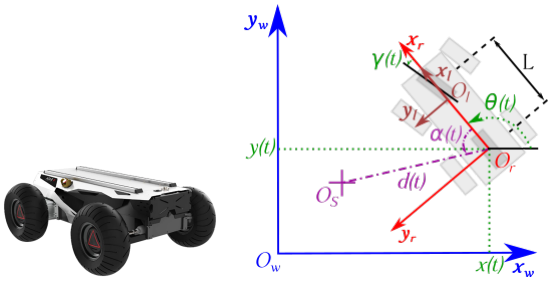
The paper is organized as follows. In the next section, we present the robotic system and the orchard environment. Next, we present the proposed navigation framework and detail the data extraction, the NMPC control and the task switching method. Finally, simulation results are presented.

[1]Univ. de Toulouse, CNRS, UPS, Toulouse, France
[2]CNRS, LAAS, 7 avenue du colonel Roche, F-31400 Toulouse, France {avillemaze, cadenat}[at]laas.fr
[3]Departamento de Engenharia de Mecânica, Universidade Federal de Pernambuco UFPE, Recife, PE, Brazil adrien.durandpetiteville[at]ufpe.br

II. MODELLING

This section is devoted to the problem modelling. It presents the considered robotic system together with the orchard and introduces the required elements to build the navigation strategy.

A. Robot modelling



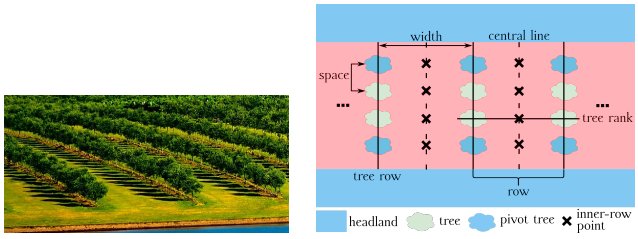
(a) Agilex Hunter 2.0 robot. (b) Robot model.

Fig. 1. The robotic system.

In this work we consider the Agilex Hunter 2.0 car-like robotic platform (see Fig. 1(a)). It is equipped with a 270° field of view laser range-finder positioned at its front allowing to detect the surroundings. To model this system, let us define the following frames (see Fig. 1(b)): $F_w = (O_w, \mathbf{x_w}, \mathbf{y_w}, \mathbf{z_w})$ fixed to the world, $F_r = (O_r, \mathbf{x_r}, \mathbf{y_r}, \mathbf{z_r})$ attached to the robot, and $F_l = (O_l, \mathbf{x_l}, \mathbf{y_l}, \mathbf{z_l})$ linked to the laser. The robot pose is given by $\chi(t) = [x(t), y(t), \theta(t)]$ where $x(t)$ and $y(t)$ are the coordinates of O_r in F_w, whereas $\theta(t)$ is the angle between $\mathbf{x_w}$ and $\mathbf{x_r}$. The control vector is defined as $\mathbf{U}(t) = [\upsilon(t), \gamma(t)]$ where $\upsilon(t)$ is the linear velocity along $\mathbf{x_r}$ and $\gamma(t)$ the angular position of the steering angle. For such a system, with L the distance between the front and rear wheel axes, the kinematic model is given by:

$$\begin{cases} \dot{x}(t) = \upsilon(t)\cos(\theta(t)) \\ \dot{y}(t) = \upsilon(t)\sin(\theta(t)) \\ \dot{\theta}(t) = \frac{\upsilon(t)}{L}\tan(\gamma(t)) \end{cases} \quad (1)$$

B. Orchard description



(a) Orchard example. (b) General structure of an orchard.

Fig. 2. The orchard.

An orchard is shaped by the hand of man. Therefore, it is an environment with a known general structure. Indeed, as shown in Fig. 2, the trees are planted along parallel straight lines, forming rows where the robot can navigate. The rows width and the distance between two consecutive trees are considered to be constant over the whole orchard and are supposed to be roughly known. To navigate through the orchard, the robot has to follow a row, switch to the next one and repeat these two motions until the area is covered. In this work, it is proposed that the first task is performed by following a row central line defined by a set of points, named inner-row points. To realize the row switching, the robot has to perform maneuvers in the headlands (the free spaces in blue on Fig. 2). To do so, it is proposed to define this task using the positions of the trees located at the rows extremity, named pivot trees. Both these latter and the inner-row points will be deduced from the laser-range data, leading to a full sensor-based approach and thus a better execution robustness. These aspects are detailed in the next section.

III. NAVIGATION FRAMEWORK

This section presents the core of our contribution. It successively describes the proposed sensor-based navigation strategy, the extraction of the required information from the sensory data and the design of a suitable controller.

A. Approach presentation

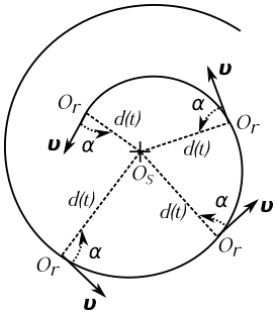


Fig. 3. Spiral model.

To design the orchard navigation strategy, two cases are considered: (i) in-row navigation, i.e., crossing from the entrance to the exit of an alley and (ii) headland navigation, i.e., driving from the current alley to the next one. In order to mathematically define these two tasks, we rely on the spiral method presented in [16]. As shown in Fig. 3, the position of a point O_s in the robot frame F_r can be expressed by its polar coordinates $\mathbf{O_s}(t) = (\alpha(t), d(t))$, where $\alpha(t)$ is the oriented angle from the $\mathbf{x_r}$ vector to the $\mathbf{O_rO_s}$ one, and $d(t)$ the distance between the points O_r and O_s. It is shown in [16] that if both $\upsilon(t)$ and $\alpha(t)$ are constant then O_r describes a spiral, and the following equation holds:

$$\dot{d}(t) = -\upsilon\cos(\alpha) \quad (2)$$

Thus, (2) shows that the type of spiral only depends on parameter α. Indeed, if $\alpha \in [-\pi; 0]$, then O_r turns clockwise with respect to O_s and counter-clockwise if $\alpha \in [0; \pi]$. Moreover, if $\alpha \in]-\pi; -\frac{\pi}{2}[\cup]\frac{\pi}{2}; \pi[$, then the spiral is outward and inward if $\alpha \in]-\frac{\pi}{2}; 0[\cup]0; \frac{\pi}{2}; \pi[$. It becomes a circle if $\alpha = \pm\frac{\pi}{2}$, with a

radius equal to d. Finally, if $\alpha = 0$, then O_r follows a straight line towards O_s or away from it if $\alpha = \pi$.

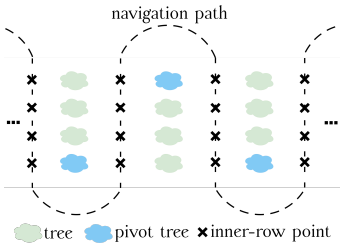


Fig. 4. Example of navigation in the orchard.

This analysis allows to design the two previously mentioned tasks. For the first one, *i.e.,* in-row navigation, it is proposed that the robot follows a path defined by a set of inner-row points (see Fig. 4). To do so, we define $\alpha^* = 0$ and $d^* = 0$ where α^* and d^* are the desired values respectively for $\alpha(t)$ and $d(t)$ for the considered inner-row point. Thus, regulating $\alpha(t)$ and $d(t)$ towards the desired values α^* and d^* allows positioning the robot at a given position O_s. To cross the alley, the robot must therefore perform a sequence of regulations towards a point O_s defined for each regulation process by the inner-row point immediately facing the robot. For the second task, *i.e.,* maneuvering in the headland, it is proposed to follow a circle centered on a pivot tree (see Fig. 4). To do so, following the analysis of (2), we impose $\alpha^* = \pm\pi/2$ and $d^* = r_c$ where r_c is the radius of the circle.

The adopted approach being stated, we now present the different techniques allowing to implement this solution. First, we detail the data processing methods used to compute the points of interest, *i.e.* the inner-row points and the position of the last trees in the row. Next, we introduce the sensor-based NMPC scheme intended to regulate the current $\alpha(t)$ and $d(t)$ towards their desired values α^* and d^*. Finally, we present our strategy to deal with the sequence of tasks.

B. Data processing

The proposed navigation strategy relies on the position of the trees in the robot frame. These data are computed from the point-cloud provided by the on-board laser-range finder. The points are first clustered using a distance criterion in order to gather the ones related to the same tree. Next, the position of each tree is calculated by averaging each cluster and expressed using polar coordinates in F_r.

Now that the positions of the surrounding trees are computed, it is necessary to determine the lines Δ_L and Δ_R passing through the left and right tree rows. To do so, we rely on the normal parametrization proposed in [17]:

$$\rho = d_i \cos(\alpha_i - \beta) \tag{3}$$

where (α_i, d_i) are the polar coordinates of a tree T_i in F_r, ρ the algebraic shortest distance from O_r to the line and β the angle from $\mathbf{x_r}$ to the line normal (see Fig. 5).

First we consider that both tree rows are straight and parallel, *i.e.,* their models share the same value for β while

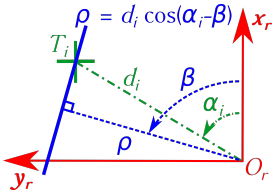
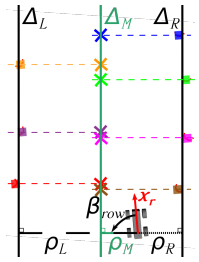


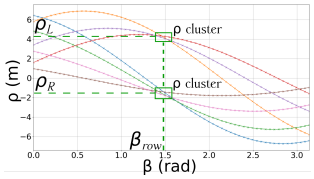
Fig. 5. Normal parametrization of a line passing through the tree T_i.

ρ has two distinct values. Thus, by varying β from 0 to π on the complete set of positions, the row orientation β_{row} is found when only two values of ρ are obtained. However, the trees are never perfectly aligned and it is then impossible to obtain only two distinct values of ρ. Thus, in order to find the parameters of the two line models, it is necessary to look for two clusters of ρ in place of two distinct values (see Fig. 6(b) and 6(d)). Finally, we have to consider one special case for the line parameters computation: mismatches at the end of the rows when four or less trees are detected. In such cases, there are two values β for which there exists two distinct values of ρ after clustering. We select the value of β for which the inter-clusters distance is the closest one to the row width[1] (see Fig. 6(c) and 6(d)).

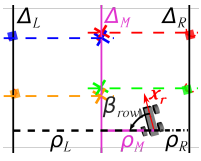
It is now necessary to compute Δ_M, the central line to Δ_L and Δ_R. ρ_m is obtained by averaging ρ_L and ρ_R and β_M is equal to β_{row} to be parallel to both lines. Finally, the inner-row points are obtained by orthogonally projecting the trees on Δ_M (see Fig. 6(a) and 6(c)). Points relative to the same tree rank are gathered when they are too close.



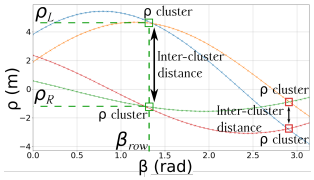
(a) 3D situation in the row.



(b) Related (ρ, β) plane in the row.



(c) 3D situation at the end of the row.



(d) Related (ρ, β) plane at the end of the row

Fig. 6. Examples of result for data processing. For the two presented cases, *i.e.* in the row and at its end, the color relies a tree position to its corresponding inner-row point and to its projection on the (ρ, β) plane.

At this step, the positions of the inner-row and of the pivot points have been computed from the laser range finder data.

[1]The special case where the row width and tree distance are equal is not taken into account in this work.

We now present how those information are used by the NMPC scheme to allow the robot to navigate in the orchard.

C. Nonlinear Model Predictive Control

In order to achieve the two presented tasks, *i.e.,* in-row and headland navigation, we rely on NMPC and we consider the following optimization problem:

$$\overline{\mathbf{U}}^*(k) = \min_{\overline{\mathbf{U}}(k)}(J_{N_p}(\mathbf{O}_s(k), \overline{\mathbf{U}}(k))) \tag{4}$$

with

$$J_{N_p}(\mathbf{O}_s(k), \overline{\mathbf{U}}(k)) = \sum_{p=k+1}^{k+N_p-1} [\hat{\mathbf{O}}_s(p) - \mathbf{O}_s^*]^T [\hat{\mathbf{O}}_s(p) - \mathbf{O}_s^*] \tag{5}$$

subject to

$$\hat{\mathbf{O}}_s(p+1) = f(\hat{\mathbf{O}}_s(p), \mathbf{U}(p)) \tag{6a}$$

$$\hat{\mathbf{O}}_s(k) = \mathbf{O}_s(k) \tag{6b}$$

$$C(\overline{\mathbf{U}}^*(.)) \leq 0 \tag{6c}$$

Thus NMPC consists in computing an optimal control sequence $\overline{\mathbf{U}}^*(k)$ of $\overline{\mathbf{U}}(k)$, with $\overline{\mathbf{U}}(k) = [\mathbf{U}(k), ..., \mathbf{U}(k+N_P-1)]$, that minimizes the cost function J_{N_p} over a prediction horizon of N_p steps while taking account the set of user-defined constraints $C(\overline{\mathbf{U}}^*(k))$. The optimal control sequence is of length N_c, which represents the control horizon. Thus, if $N_p > N_c$, then the last $N_p - N_c$ predictions are all obtained using the last control input of $\overline{\mathbf{U}}^*(k)$.

Cost function: The cost function is the sum of the quadratic difference between $\hat{\mathbf{O}}_s(.)$, the predicted polar coordinates $(\hat{\alpha}, \hat{d})$ of a given point of interest in a set of N_p predicted robot frames $\hat{F}_r(.)$, and $\mathbf{O}_s^*$ the desired coordinates of a point O_s in the robot frame. These latter are equal to $(0,0)$ to make the robot cross the alley, and to $(\pm\frac{\pi}{2}, r_c)$ to drive the robot in the headland.

Prediction model: In order to predict the polar coordinates $\hat{\mathbf{O}}_s(p+1)$ as shown in (6a), we proceed as follows. First, for a command vector of the given sequence $\overline{\mathbf{U}}(k)$ of N_p commands, we compute the two dimensional homogeneous transformation matrices $H_{p|p+1}$ such as

$$H_{p|p+1} = \begin{bmatrix} R_{p|p+1} & T_{p|p+1} \\ 0_{1\times 2} & 1 \end{bmatrix} \tag{7}$$

where $R_{p|p+1}$ and $T_{p|p+1}$ are respectively the rotation matrix and translation vector between $\hat{F}_r(p)$ and $\hat{F}_r(p+1)$, with $p \in [1, ..., N_p]$. They depend on $\Delta x_{p|p+1}$, $\Delta y_{p|p+1}$ and $\Delta\theta_{p|p+1}$, respectively the displacement along the $\mathbf{x}_r$ and $\mathbf{y}_r$ axis, and the rotation around the $\mathbf{z}_r$ axis between the frames $\hat{F}_r(p)$ and $\hat{F}_r(p+1)$. $\Delta x_{p|p+1}$, $\Delta y_{p|p+1}$ and $\Delta\theta_{p|p+1}$ can be obtained by integrating the kinematic model given by (1) over a sampling time T_s and considering a constant velocity vector $\mathbf{U}(p)$. Next, we deduce the homogeneous Cartesian coordinates $\hat{\mathbf{O}}_s^c(p)$ from its polar ones $\hat{\mathbf{O}}_s(p)$. It is then possible to predict the homogeneous Cartesian coordinates of O_s in the predicted frame $F_r(p+1)$ as follows:

$$\hat{\mathbf{O}}_s^c(p+1) = H_{p|p+1}^{-1} \hat{\mathbf{O}}_s^c(p) \tag{8}$$

Finally, from $\hat{\mathbf{O}}_s^c(p+1)$ we deduce $\hat{\mathbf{O}}_s(p+1)$. This process is initialized using (6b) and is repeated for the N_p steps.

Zero terminal equality constraint: The stability of the NMPC schema on the finite horizon N_p is obtained by adding a zero terminal equality constraint. The respect of this constraint at each iteration guarantees the recursive feasibility, *i.e.*, there exists a trajectory from the current state leading to the desired one. It is defined as the error, in F_r, between the Cartesian coordinates $(\hat{x}(.), \hat{y}(.))$ of a predicted position of the robot $\hat{\phi}(.)$ and the desired one ϕ_{ZEC}. The zero equality constraint is thus expressed as:

$$\|\hat{\phi}(.) - \phi_{ZEC}\| = 0 \tag{9}$$

In practice, the equality constraint is impossible to achieve and the following inequality constraint is used instead:

$$\|\hat{\phi}(.) - \phi_{ZEC}\| - \delta_{ZEC} \leq 0 \tag{10}$$

where δ_{ZEC} is a user-defined threshold sufficiently small to offer an efficient implementation of the constraint while impacting the optimisation process as the equality constraint.

ϕ_{ZEC} is different for each task. When regulating towards a point, ϕ_{ZEC} is set to the Cartesian coordinates of O_s to force the robot to reach the point. When maneuvering in the headland, ϕ_{ZEC} is computed as the Cartesian coordinates of a point on the desired spiral. This allows the robot to keep moving forward even after reaching (α^*, d^*).

Input constraints: The constraints applied to the control inputs allow to take into account the physical limits of the actuators. Moreover, for the set of predictions, they bound the optimization problem. They are defined as:

$$\begin{bmatrix} \mathbf{U}(i) - \mathbf{U}_u \\ \mathbf{U}_l - \mathbf{U}(i) \end{bmatrix} \leq 0 \tag{11}$$

where $i \in [1, N_c]$, $\mathbf{U}_l$ and $\mathbf{U}_u$ the lower and upper boundaries.

D. Task switching

The presented control scheme allows to achieve two tasks: tracking a 'flat spiral' to reach a given position or tracking a 'circle-shape spiral' for the headland navigation. First, to cross an alley, it is necessary to perform position regulations one after the other with respect to a sequence of inner-row points. Second, the robot has also to switch from the sequence of position regulation tasks to the circle-shape tracking one, and *vice-versa*. Thus, there exists threes cases for switching: (i) from position regulation to position regulation, when the robot crosses the alley, (ii) from position regulation to circle-shape tracking, when the robot leaves the alley and enters the headland, and (iii) from circle-shape tracking to position regulation, when the robot leaves the headland and enters the next alley (see Fig. 4). To make the robot switch from one task to another one, it is necessary to update $\mathbf{O}_s$, α^* and d^*. We now detail how the switch is performed for the three previously mentioned cases.

For the first switching case, the robot is initially regulating its position with respect to the first inner-row point in front of it, here referred to as O_1, and the N_p predictions are function

of the predicted polar coordinates of O_1, $\alpha^* = 0$ and $d^* = 0$. While the robot drives towards O_1, the predicted trajectory gets shorter and the n_z last predicted positions, with $n_z \geq 1$, are equal, or extremely close, to the desired position. In such a case, these n_z predictions, which are no more useful to reach the desired position O_1, are now used to track O_2, the second inner-row point in front of the robot. In other words, the n_z last terms of the cost function are now function of the predicted polar coordinates of O_2, $\alpha^* = 0$ and $d^* = 0$. If necessary, the cost function is modified at each iteration, up to obtaining a cost function only depending of the coordinates of O_2. In that case, O_2 is now the main reference and the process is repeated with O_3, the next inner-point in front of the robot, until the end of the row is reached.

For the second switching case, the robot is first navigating with respect to the last inner-row point of the alley. Similarly to the first case, when the robot begins to be close enough to the point, the n_z last predictions are equal, or extremely close, to the desired position. In this case, these predictions are used to track the circle-shape spiral centered on the pivot tree. In other words, the n_z last terms of the cost function are now function of the predicted polar coordinates of the pivot tree, $\alpha^* = \pm\pi/2$ and $d^* = r_c$. If necessary, the cost function is modified at each iteration, up to obtaining a cost function only depending of the coordinates of the pivot tree.

For the last case, the robot is initially tracking the circle-shape spiral with a terminal constraint with respect to a moving point ϕ_{ZEC} belonging to the spiral. When the first inner-row point of the next alley, here referred to as O_1, becomes visible, two changes are made. First, r_c is modified to make the circle pass through O_1. Indeed, as the trees are not perfectly aligned and spaced, it is necessary to adjust the circle radius to make it pass through the inner-row point. O_1 is now candidate to become the terminal constraint point ϕ_{ZEC}. As soon as O_1 is closer to the robot than the current ϕ_{ZEC}, it becomes the terminal constraint point ϕ_{ZEC}. This point defines the end of the circle-shape spiral following and it is now possible to rely on a switching strategy similar to the two other cases.

IV. SIMULATION

A. Simulation environment and parameters

The simulation robot is a car-like mobile base inspired by the Hunter 2.0 robot (see Fig. 1(a)). Its wheelbase L is set to 0.65 meter, while the limits on the control inputs are defined as follows: $v(t) \in [0, v_M]$ with $v_M = 1$ m/s and $\gamma(t) \in [-\gamma_M, \gamma_M]$ with $\gamma_M = 0.69$ rad. Finally, the embedded laser range finder is located at the robot front, providing the necessary data to detect the surrounding trees. Two examples of data acquisition are shown in Fig. 7 and Fig. 8.

The orchard is composed of 4 rows made of 10 trees. The rows are 20 meters long and 6 meters apart, whereas the space between two successive trees is fixed to 2 meters. A randomisation was applied on the tree positions (about 10 centimeters) and on their orientations (about 360 degrees). The robot has to drive through the first alley, turn on the left to reach the next alley, cross the second alley, turn to its right and finally drive through the last alley. The direction the robot has to turn at the end of an alley are given.

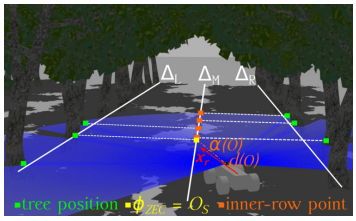


Fig. 7. Data acquisition during the in-row navigation.

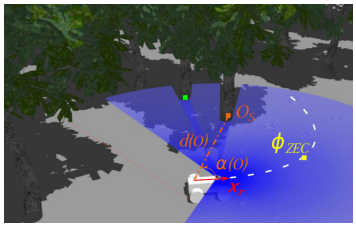


Fig. 8. Data acquisition during the headland navigation.

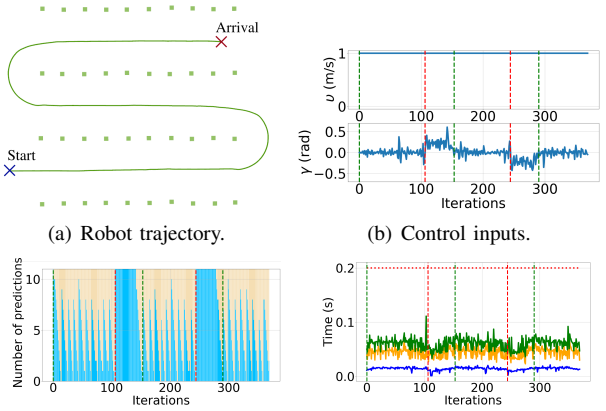
The presented simulation is implemented with ROS and coupled with Gazebo. The NMPC is implemented using the Python 3.8 language and the cost function is minimised with the SQP solver from the NLopt package [18].

Now, concerning the simulation parameters themselves, the sampling period has been fixed to $T_s = 0.2$ second, and both the prediction and control horizons have been taken equal to 12. Moreover, for the parameters of the NLopt, the stopping criteria are the absolute tolerance on the cost function and the maximum optimization time respectively set to 0.05 and 0.16 second, the zero equality constraint threshold $\delta_{ZEC} = 0.006$ and the boundaries of optimization problem are similar to the control inputs ones.

B. Simulation results

The obtained results[2] are presented in Fig. 9 where the green vertical dashed lines represent the beginning of an in-row navigation and the red ones the beginning of the maneuver in the headland. In Fig. 9(a) we can see that the robot manages to achieve the navigation task by crossing the three alleys and maneuvering in the two headland sections, driving a total of 73.14 meters in 75.2 seconds. Regarding the control inputs, the linear velocity computed by the optimization process remains at its maximal value throughout the whole navigation (see Fig. 9(b)). This is mostly due to the inclusion of the zero terminal constraint that forces the robot to move forward to reach the desired position. Regarding the steering angle, it tends to 0 rad during the in-row navigation and to 0.2 rad during the headland maneuvers, which is the expected value to describe a circle of approximately 3 meters (width / 2) (Fig. 9(b)). In Fig. 9(c), the distribution of the interest points,

[2]A video is also available https://youtu.be/YMKyckTEpSA



(a) Robot trajectory.

(b) Control inputs.

(c) Distribution of the points of interest. The blue and orange bars denotes respectively the number of predictions associated with the first and second point in front of the robot.

(d) Processing times. Blue line: data processing time - Orange line: NLopt solving time - Green line: total processing time - Red dotted line: T_s.

Fig. 9. Simulation results.

i.e., the tasks, is shown. During the in-row navigation, the number of predictions assigned to the first inner-row point decreases from its maximal value to zero. The second inner-row points then becomes the first point of interest and the process is repeated up to the end of the alley. Before entering the headland (before the red dashed lines), we can see that the pivot tree is progressively included as the second point of interest. Once the robot starts the maneuver (red dashed lines), the whole prediction set remains exclusively dedicated to the circle tracking for a large period of time. It is only when the first inner-row point can be calculated (before the green dashed lines) that it is progressively included as the second point of interest. When the robot enters the next alley (green dashed lines) the transition on the prediction set is completed and it only contains the first inner-row point as point of interest. It should be noticed that despite the numerous switches between the tasks, the robot manages to achieve the navigation task. Finally, the processing times given in Fig. 9(d) show that the proposed approach is compatible with a robotic application. Indeed, the total processing time is lower than 100 ms, except for a couple of peaks that could be managed with a time out, offering a 10 hz command rate. Moreover, the solver does not seem to be greatly disturbed by the numerous changes on the optimization when switching tasks.

V. CONCLUSION

We have presented a navigation strategy relying on sensor-based model predictive control to make a robot autonomously navigate in an orchard. The proposed solution extends previous works and offers a sole framework based on spirals to deal the same way with both in-row and headland navigation tasks. It also makes possible the task switching by an adequate update of the spiral parameters and without risking to destabilize the robot. Finally, the design of a sensor-based NMPC instead of more classical non linear control law allows to explicitly deal with several constraints such as stability and actuator saturation. The obtained simulation results show the relevance and the efficiency of the approach.

For future works, we plan to experiment this solution on the Agilex Hunter 2.0 mobile base and test it in real orchards. We aim also at benefiting from the NMPC control structure to integrate new constraints allowing to avoid obstacles, thus performing more complex navigation tasks.

REFERENCES

[1] J. A. Foley, N. Ramankutty, K. A. Brauman, E. S. Cassidy, J. S. Gerber, M. Johnston, N. D. Mueller, C. O'Connell, D. K. Ray, P. C. West, *et al.*, "Solutions for a cultivated planet," *Nature*, vol. 478, no. 7369, p. 337, 2011.

[2] J. F. Reid, "The impact of mechanization on agriculture," *Bridge*, vol. 41, no. 3, pp. 22–29, 2011.

[3] M. Li, K. Imou, K. Wakabayashi, and S. Yokoyama, "Review of research on agricultural vehicle autonomous guidance," *International Journal of Agricultural and Biological Engineering*, vol. 2, no. 3, pp. 1–16, 2009.

[4] J. Radcliffe, J. Cox, and D. M. Bulanon, "Machine vision for orchard navigation," *Computers in Industry*, vol. 98, pp. 165–171, 2018.

[5] S. Opiyo, C. Okinda, J. Zhou, E. Mwangi, and N. Makange, "Medial axis-based machine-vision system for orchard robot navigation," *Computers and Electronics in Agriculture*, vol. 185, p. 106153, 2021.

[6] A. Durand-Petiteville, E. Le Flecher, V. Cadenat, T. Sentenac, and S. Vougioukas, "Tree detection with low-cost three-dimensional sensors for autonomous navigation in orchards," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3876–3883, 2018.

[7] J. Gai, L. Xiang, and L. Tang, "Using a depth camera for crop row detection and mapping for under-canopy navigation of agricultural robotic vehicle," *Computers and Electronics in Agriculture*, vol. 188, p. 106301, 2021.

[8] P. M. Blok, K. van Boheemen, F. K. van Evert, J. IJsselmuiden, and G.-H. Kim, "Robot navigation in orchards with localization based on particle filter and kalman filter," *Computers and Electronics in Agriculture*, vol. 157, pp. 261–269, 2019.

[9] A. Danton, J.-C. Roux, B. Dance, C. Cariou, and R. Lenain, "Development of a spraying robot for precision agriculture: An edge following approach," in *2020 IEEE Conference on Control Technology and Applications (CCTA)*. IEEE, 2020, pp. 267–272.

[10] S. G. Vougioukas, "Agricultural robotics," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 2, pp. 365–392, 2019.

[11] V. Subramanian and T. F. Burks, "Autonomous vehicle turning in the headlands of citrus groves," in *2007 ASAE Annual Meeting*. American Society of Agricultural and Biological Engineers, 2007, p. 1.

[12] G. Bayar, M. Bergerman, A. B. Koku, and E. Ilhan Konukseven, "Localization and control of an autonomous orchard vehicle," *Computers and Electronics in Agriculture*, vol. 115, pp. 118–128, 2015.

[13] J. Zhang, S. Maeta, M. Bergerman, and S. Singh, "Mapping orchards for autonomous navigation," in *2014 Montreal, Quebec Canada July 13–July 16, 2014*. American Society of Agricultural and Biological Engineers, 2014, p. 1.

[14] A. Durand-Petiteville, E. Le Flecher, V. Cadenat, T. Sentenac, and S. Vougioukas, "Design of a sensor-based controller performing u-turn to navigate in orchards," in *International Conference on Informatics in Control, Automation and Robotics*, vol. 2, 2017, pp. 172–181.

[15] E. Le Flecher, A. Durand-Petiteville, F. Gouaisbaut, V. Cadenat, T. Sentenac, and S. Vougioukas, "Nonlinear output feedback for autonomous u-turn maneuvers of a robot in orchard headlands," in *International Conference on Informatics in Control, Automation and Robotics (ICINCO)*, 2019.

[16] K. N. Boyadzhiev, "Spirals and conchospirals in the flight of insects," *The college mathematics Journal*, vol. 30, no. 1, p. 23, 1999.

[17] P. E. H. R. O. Duda, "Use of the hough transformation to detect lines and curves in pictures," *Communications of the ACM*, vol. 15, no. 1, pp. 11–15, 1972.

[18] S. G. Johnson, "The nlopt nonlinear-optimization package," 2020. [Online]. Available: http://github.com/stevengj/nlopt